Executable Calling Signatures for FRVT 2006 [modified 06-01-18]

1-1 Matching:

executable ***parameter_file image_directory target_sigset query_sigset similarity_file quality_target_sigset quality_query_sigset***

1-Many Matching:

executable ***parameter_file image_directory target_sigset query_sigset similarity_file quality_target_sigset quality_query_sigset***

Similarity Score Normalization:

executable ***parameter_file similarity_file normalized_similarity_file probe_sigset gallery_sigset***

Preprocessing:

executable ***parameter_file image_directory input_sigset output_image_directory output_sigset***

All Duplicate Detection:

executable ***parameter_file image_directory target_sigset dup_list***

Executable Arguments

Table A describes the arguments used above in the calling signatures for the executables. It is important to note that all filenames will be relative to the working directory from which the program is instantiated.

Parameter name	Type	Format	Description
parameter_file	Input	XML	An XML document that specifies experiment description information, configuration parameters and the name of metadata files.
image_directory	Input	string	The relative path to the image (data) directory
output_image_directory	Output	string	The relative path to the directory where preprocessed images should be written.

Parameter name	Type	Format	Description
target_sigset	Input	Sigset	The name of the target signature set. This document will contain a list of the target images.
query_sigset	Input	Sigset	The name of the target signature set. This document will contain a list of the query images.
similarity_file	Input/ Output	Similarity Matrix	The name of the similarity file. For the 1:1 and 1:many matching experiments, this will be the primary output data structure. This will be an input data structure for the similarity normalization experiment.
quality_target_sigset	Output (optional)	Sigset	The name of the target sigset to which quality scores should be written. This file should be a copy of the input target sigset with quality scores added. Ignore this parameter if your executable does not compute quality scores. The quality score option is only available for 1-1 matching, not for 1-many matching. For 1-many matching a filename will be provided to be consistent with the calling signature.
quality_query_sigset	Output (optional)	Sigset	The name of the query sigset to which quality scores should be written. This file should be a copy of the input query sigset with quality scores added. Ignore this parameter if your executable does not compute quality scores. The quality score option is only available for 1-1 matching, not for 1-many matching. For 1-many matching a filename will be provided to be consistent with the calling signature.
normalized_similarity_file	Output	Similarity Matrix	The name of the normalized similarity matrix. The dimensions of the similarity matrix should correspond to the size of the gallery and probe set. In the header, the referencing target sigset should be the gallery sigset, and the referencing query sigset should be the probe sigset.
probe_sigset	Input	Sigset	The name of the probe sigset. This file will contain a list of probe images to use during normalization.
gallery_sigset	Input	Sigset	The name of the gallery sigset. This file will contain a list of gallery images to use during normalization.
input_sigset	Input	Sigset	The name of the file that contains the list of image for preprocessing.
output_sigset	Output	Sigset	The name of the file to which the processed sigset is written. This file should contain the name of the processed images. This list should have the same number of images as the

Parameter name	Type	Format	Description
			input_sigset.
dup_list	Output	Duplist	The name of the file to which the duplicate list is written.

Table A Descriptions of executable arguments

Directory Structure

Each participant will be given the directory structure shown in Figure A. All experiments (and thus executables) will be instantiated from the top level (*/home*) directory. Executables must be installed to the */bin* directory. Any libraries needed by executables should be installed in the */lib* directory. After installation, both the */bin* and */lib* directory will be made read and execute only. All required outputs (similarity matrices, signature sets, duplicate lists) generated during experiment execution should be written to the */output* directory. Users will be given write permission to the */output* directory. The */temp* directory should be used for any temporary files generated during program execution. Users will be given read and write privileges to the /temp directory. It is important to note that files output to the temporary directory may be deleted prior to subsequent runs of experiments. Images will be stored outside of the directory structure in Table A. Users will be given the relative path to the images via the *image_directory* parameter.

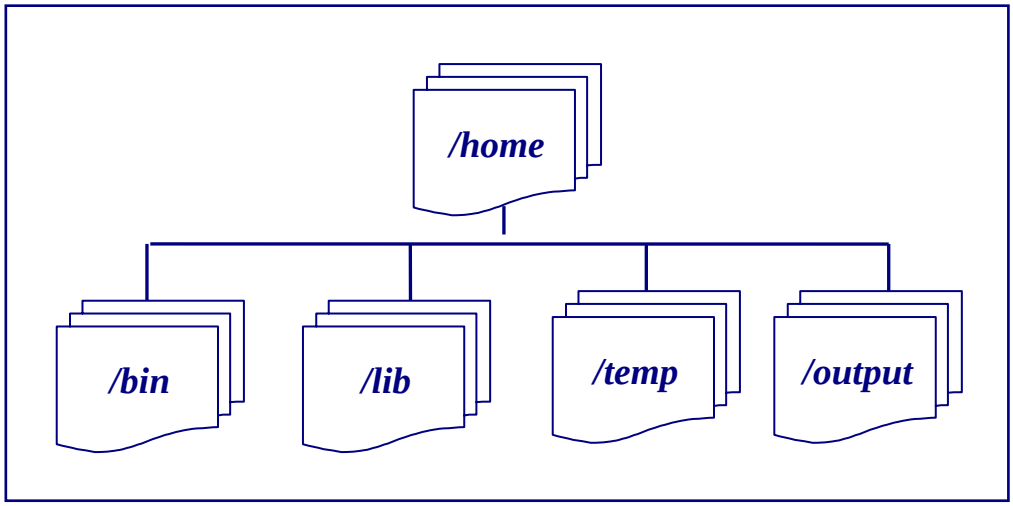

Figure A FRVT 2006 directory structure.

Image Directory

The image directory is a string that specifies the relative path to images. This string should be appended to each of the data files listed in the Signature Sets. For examples, if the parameter *image_directory* is '..\images\' and the Sigset refers to a image named '2D\image1.jpg', you should process the image in the file '..\images\2D\image1.jpg'. Note: the value of the *image_directory* argument will be consistent with the underlying operating system. Thus, '..\images\' and '../images/' would be provided for windows and Linux operating system respectively.

Parameter Files

Overview

Parameter files are XML documents that provide experiment description information and configuration values for executables. To simplify their processing, configuration values will always be specified via the values of attributes in the parameter file.

Structure

While the precise content of the Parameter files has not been determined, the structure will be similar to the example shown below in

Figure **B**. In this example, elements are depicted blue (capitalized), attributes are red (lowercase) and attribute values are black (uppercase and in quotes). The *Experiment* element is the outer element. It has one attribute named *segmentation* with value "Automatic". The *Experiment* element has two child elements *Target* and *Query*. Both the *Target* and *Query* elements have four attributes: *mode*, *capture*, *min_recordings* and *max_recordings*.

```
<?xml version="1.0"?>
<Experiment name="1_to_Many" type="1-many" feature_extraction_mode="Full"
ground_truth="FRVT2006metadata.xml" >
 <Target mode="3D" capture="Controlled" min_recordings="1" max_recordings="3" pose="Frontal" />
 <Query mode="2D" capture="Uncontrolled" min_recordings="1" max_recordings="1"
pose="Nonfrontal" />
 <LogFile name="1_to_many_logfile.txt" />
</Experiment>
```

Figure B Example of a parameter file

Attribute	Type	Allowed Values	Comments
name (Experiment)	String		The text of this string provides a name for the experiment. This attribute is not needed by users.
Type	String	1-1 1-many AllDuplicate Preprocessing Normalization	The type of task and executable that this is experiment is being evaluating.
feature_extraction_mode	enumerated	Partial Full	The enumerated values of this attribute specify whether ground truth data is provided to aid in feature extraction. “Partial” indicates that ground truth is provided in the file denoted by the attribute *ground_truth*. “Full” indicates that ground truth data is not provided.
Mode	enumerated	2D 3D	The enumerated values of this attribute specify whether the images are 2D or 3D face images.
Capture	enumerated	Controlled Uncontrolled	The enumerated values of this attribute specify whether the images were captured under controlled or uncontrolled conditions.
min_recordings	integer	>0	This integer specifies the minimum number of images that will be associated with a signature.
max_recordings	integer	>0	This integer specifies the maximum number of images that will be associated with a signature.
Pose	string	Frontal Nonfrontal	Pose variation of the biometric signatures in the sig-set.
Name (LogFile)	string	Filename	All logging information generated by the program should be written to this file in the output directory.

Table B Description of elements in the similarity header.

Parsing

Due to their simple structure, Parameter files are readily parsed with any XML or XPath parser. Source implementation of C++ and Java classes for parsing Parameter files are provided in the BEE (Biometric Experimentation Environment) distribution. These classes, which use the XPath parser, are available to FRVT 2006 participants. Parsers will not be made available for other languages (e.g. Matlab). However, users should be able to easily create parsers in other languages using the C++ classes as a guide.

Writing

FRVT users will not be required to write (output) Parameter files.

Signature Sets (Sigsets)

Overview

Signature Sets (Sigsets) are the primary input structure for FRVT 2006. They will use used to list the files in the Target and/or Query sets. They will also be an output format created by preprocessing executables.

Structure

The Signature Set document will provide a list of images. XML will be used because its hierarchical structure facilitates a flexible representation of the relationships between subjects, sessions, sensors and files. Specifically, the Signature Set will consist of a list of *Signature* (subjects) element. Each *Signature* element will contain one or more *Presentation* child elements that correspond to capture sessions. Each *Presentation* element will contain one or more *Component* elements that correspond to a sensor. Lastly, each *Component* will have one or more *Data* elements that correspond to a file.

Figure C illustrates the general Signature Set structure with elements depicted blue, attributes depicted red (lowercase) and attribute values depicted black (uppercase and in quotes). This example has two **complex-biometric-signature** elements and thus represents two subjects. (Note: While **presentation-components** grouped under a single **complex-biometric-signature** are guaranteed to correspond to the same subject, separate signatures may correspond to the same subject.) The signatures are named “10928” and “10929”. Signature “10928” has two **presentation-component**s (capture sessions) labeled “10928A” and “10928B” respectively. Presentation “10928A” consists of a single 2D facial image stored in the JPEG file

"*data/02463.jpg*". Presentation "10928B" consists of a 3D face image store as a JPEG textual map in file "*data/02464.jpg*" and a shape map store in ABS format in file "*data/02465.jpg*". Signature "10929" has a single **presentation-component** that corresponds to a single 2D facial image stored in the JPEG file "*data/02466.jpg*".

```
<?xml version="1.0" encoding="UTF-8"?>
<biometric-signature-set  xmlns="http://www.bee-biometrics.org/schemas/sigset/0.1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.bee-biometrics.org/schemas/sigset/0.1
  http://www.bee-biometrics.org/schemas/sigset/0.1/general.xsd">

 <complex-biometric-signature name="10928" >
  <complex-presentation name="10928A" modality="face" >
   <presentation-component name="10928A_2D_face" >
    <data file-name="data/02463.jpg" file-format="jpeg" />
   </presentation-component>
  </complex-presentation>
  <complex-presentation name="10928B" modality="face" >
   <presentation-component name="10928_3D_face_shape" >
    <data file-name="data/02464.jpg" file-format="jpeg" />
   </presentation-component>
   <presentation-component name="10928_3D_face_texture" >
    <data file-name="data/02465.abs" file-format="abs" />
   </presentation-component>
  </complex-presentation>
 </complex-biometric-signature>
 <complex-biometric-signature name="10929" >
  <complex-presentation name="10929" modality="face" >
   <presentation-component name="10929" >
    <data file-name="data/02466.jpg" file-format="jpeg" />
   </presentation-component>
  </complex-presentation>
 </complex-biometric-signature>
</biometric-signature-set>
```

Figure C Example of a Signature Set.

Parsing

Signature sets are difficult to parse due to their rich hierarchical structure. Fortunately, C++ and Java classes for parsing Similarity Matrices are provided in the BEE (Biometric Experimentation Environment) distribution. These classes are available to FRVT 2006 participants. We also provide examples or the use of these parsers. It is important to note that some of the BEE examples assume the simplified Sigset structure in which each Signature has precisely one Presentation, each Presentation has precisely one Component and each Component has precisely one Data member. Care should be used which using this simplified version of the Similarity

structure. Parsers are not provided for other languages (e.g. Matlab). However, users should be able to create a Java wrapper that parses the Signature Set and passed the appropriate data structures to this executables.

Writing

Like most XML documents, Signature sets are easier to write than they are to parse. Thus, users can either use the C++ and Java classes supplied in BEE to create Signature Sets or output them directly. We recommend that the supplied classes be used because they have been rigorously tested and can easily be made compliant with changes in the schema for Signature Sets.

Similarity Matrices

Overview

Similarity Matrices are the primary output structures of recognition algorithms in FRVT 2005. They consist of a header that specified the type and dimension of the contained data and the $N \times M$ scores from the biometric algorithm.

Structure

The Similarity Matrix is similar to many image files in that it contains a textual header prepended to a binary representation of a $N \times M$ data structure. The structure of the header is depicted below in Figure D. Here, we see that the header consist of four lines. The first line must contain the either the character ‘D’ (for distance matrix) or the character ‘S’ (similarity matrix) followed by the character ‘2’. The second and third lines should contain the name of the target and query signature set respectively. The target and query name should be the same as they were specified in the call to the matching executable. The fourth line should contain the characters ‘MF’, a space, the number of signatures in the query sigset, a space, the number of signatures in the target sigset, a space and the integer 0x12345678 written in binary format. All four lines in the header should be terminated by an end-of-line character. Table C describes the elements in the similarity header.

D2
BEE_DIST/signature_sets/ICE_Exp_1.0.1_Target.xml
BEE_DIST/signature_sets/ICE_Exp_1.0.1_Query.xml
MF 1425 1425 xV4

Figure D Example of the Similarity Matrix header.

Name	Format	Separator	Comments
Storage Type	Character ‘S’ or ‘D’	none	Specified similarity scores, ‘S’, or distance measures ‘D’.
Version	The character ‘2’	eol	The value ‘2’ corresponds to the version of similarity matrix.
Target name	string	eol	This string should be the same as the name of the target sigset provided to the matching algorithm.
Query name	string	eol	This string should be the same as the name of the query sigset provided to the matching algorithm.
Format	The characters ‘MF’	space	The values correspond to a matrix, ‘M’, containing float, ‘F’, values.
Rows	integer	space	The number of signatures in the query set.
Cols	integer	space	The number of signatures in the target set.
Magic number	0x12345678 (4-bytes binary)	eol	This binary value is used to check for Endian (byte swapping).

Table C Description of elements in the similarity header.

The scores are written to the file immediately following the header. These should be $N \times M$ 4-byte binary floating point values. Here, N is the number of signatures in the query set and M is the number of signatures in the target set. Thus, the first M values correspond to comparing the first query image to each of the target images. There must not be any white space characters separating scores in the body of the Similarity Matrix.

Parsing

C++ and Java classes for parsing Similarity Matrices are provided in the BEE (Biometric Experimentation Environment) distribution. These classes will are available to FRVT 2006 participants. Parsers will not be made available for other languages (e.g. Matlab). However, users should be able to easily create parsers in other language using the C++ classes as guides.

Writing

C++ and Java classes for writing Similarity Matrices are provided in the BEE (Biometric Experimentation Environment) distribution. These classes are available to FRVT 2006 participants. Parsers will not be made available for other languages (e.g. Matlab). However, users should be able to easily write Similarity Matrices in other language using the C++ classes as guides.

Duplicate List (Dup List)

Overview

A duplicate (dup) list is an XML document that list signatures grouped by common signatures (subjects). Each *Group* element corresponds to a unique subject and the enclosed *Signature* elements correspond to the signatures or that subject in the corresponding signature set.

Structure

The structure of Dup List is shown below in Figure E. Here, elements are depicted blue (capitalized), attributes are red (lowercase) and attribute values are black (uppercase and in quotes). The example consists of an enclosing *Groups* element that has zero or more member elements named *Group*. Each *Group* element has an optional *name* attribute that provides a name for the group and two or more member elements named *Signature*. Each *signature* has an *id* attribute and an optional *confidence* attribute. The *id* attribute must reference a signature name in the corresponding sigset. The *confidence* attribute provides an integer score (0-100) corresponding to an user defined confidence level

```
<?xml version="1.0" encoding="UTF-8"?>
<Groups>
  <Group name=”first_group_name” >
    <Signature id=”signature_id” confidence=”85” />
    <Signature id=”signature_id” confidence=”75” />
    <Signature id=”signature_id” confidence=”100” />
  </Group>
  <Group name=”second_group_name” >
    <Signature id=”signature_id” confidence=”55” />
    <Signature id=”signature_id” confidence=”95” />
  </Group>
</Groups>
```

Figure E Example of the Dup List

Parsing

FRVT users will not be required to parse Duplicate Lists.

Writing

We will provide C++ and Java classes for writing Duplicate Lists. Currently, these classes are not available since Duplicate Lists are new to FRVT 2006. Also, participants can output Dup List directly within their programs since the Dup List structure is very simple.

Log Files

Each executable should produce a log file. The name (and directory) for the log file will be provided in the parameter file via the *name* attribute of the *LogFile* element. The log file should provide sufficient information so that we (with your limited support) can easily and quickly troubleshoot your algorithm. Since we will have limited time to troubleshoot programs, please provide detailed debug messages in your log file.